ALGORITHM AND PROGRAMMING

FINAL PROJECT

Diagram

Description automatically generated

**Student Information:**

Leonardo Richie

**Class Information:**

Class: L1BC

Lecturer’s name:

Jude Martinez

**Binus University International**

**School of Computer Science**

**2022**

**Chapter I: Introduction**

After studying the basics of python for 2 months in the algorithm and programming class, students are asked to do a final project to create anything related to python programming. It should be challenging and beyond of what the students have learned in class. It should have a PDF, Project program files and a video demo of the coding when it runs.

In this project I’ve decided to create a game by using pygames module. At the beginning I try to find the type of my game, I want to make. I’ve never made a game before using pygames so this is my first time. I prefer to do the pixel games rather than 3d. Finally, I came up with an idea to create a pixel parkour game with levels. By using classes, creating objects, indexing and pygames function, I can create different kind of block in the games and create the player movement for the game to run. First, I look how to create the screen, player movement, blocks, and other stuff as well from a tutorial and combined it with what I have learned from the basic of python. As for the images I download it from the sprite pack on the internet.

I started doing the project on 21 December 2020. I use Visual Studio Code as my IDE since its more comfortable for me.

**Chapter II: Project Specification**

**Project Purpose:**

The code is to create the world map of the parkour including the object as a person and blocks which is used to build up the world. To create this game, pygame is needed to support the coding. It is a new thing to learn besides the basic of python programming.

**Project Audience:**

People who want to know the coding behind creating a parkour game and learn to create different kind of classes.

**Project Aim:**

Create a parkour game with different levels where the player can move freely in the world. Once the player collides with an enemy or water they will stop running (died) and respawn at the first place they were spawned. The world is created by different types of blocks including the player, enemy, water, and dirt block

**Project Requirements:**

* A class for the button(reset), player, world, enemy, water, blocks (normal tile) and exit door.
* A class player which can move freely with animation and stop working when it collides with an enemy or water
* A class button to reset the player position after colliding
* A class of world to display the whole tile world
* World design for each level
* Pygame module to load the image, sound effect, display the screen, how the game operates
* Images for the game resources

**Chapter III: Solution Design**

1. **Overview**

A map of the world

Description automatically generated with low confidence A screenshot of a video game

Description automatically generated

Jungle run is a parkour game where the player must reach the exit door which is located on the top right of the world for every levels.

A picture containing clipart

Description automatically generated

Jungle Run uses Python 3.9.4and Pygame as its external libraries

Pygame is used to visualize and create the game so it can run. Starting from the display, loading images, adding sound effect, creating the player movements are all using the pygame module.

1. **Setting the game**

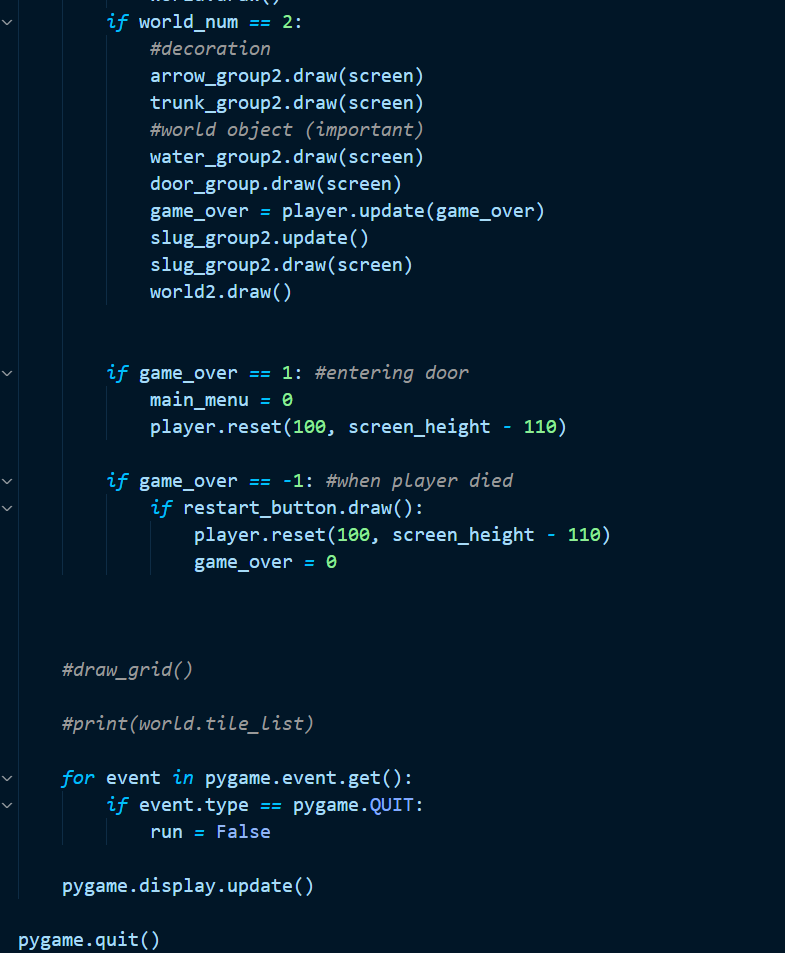
Text, chat or text message

Description automatically generated

Before creating the class and the main loop, sound effect, image, fps needs to be set. Clock = pygame.time.Clock() is to create an object that can track time. Then the fps is 60 for the game to run. To import music, pygame.mixer.music.load is used and pygame.mixer.play to play the music. Screen\_width and height are used to create the screen size. For the game, the width and the height are 1000-pixel x 1000 pixel. The pygame.display.set\_mode() is to set the screen to display the game, and by using the display.set\_caption(),the title of the game can be display on the top left caption. Tile size is to set the size for every tile in the world. For this game, the tile size is 50 which means the size of the world is 20 tiles x 20 tiles of block (1000: 50 = 20). Pygame.image.load is used to import the images in png from the file into the coding. There are many images and every each of them is put inside a variable to be used later on.

1. **The main code when game is running**

Text

Description automatically generated 

1. The while loop is used when the game is still running. When the user quit the game, run = False and the loop will stop
2. Clock.tick(fps) is used to create a fixed fps for the game. I used 60fps for my game.
3. Screen.blit(bg\_img,(0,0) create a background image (bg\_img file) starting from 0,0 coordinate starting from the top left of the screen.
4. The if statement for each main\_menu is used to change the screen to different pages.
5. Screen.blit(Rules\_img,(200,500) is to create the image rules on the front page of the game
6. Exit\_button.draw() and start\_button.draw() create buttons on screen which can be pressed(start leads to the level selection page and exit leaves the game)
7. World\_num = 1 or 2 is used to change the screen to level 1 or 2
8. .draw(screen) is used to draw different types of class (water,trunk,arrow,door,slug) which I created
9. World.draw is to draw the tile image of the world like dirt blocks
10. Game\_over = player.update(game\_over) is to set the player into the screen
11. Player,reset it used to reset the player to the starting point where they first spawned
12. Pygame.Quit is for the user to quit the game.
13. Pygame.display.update is to always display and update all the function unto the screen
14. **World design**

Chart

Description automatically generated Surface chart

Description automatically generated

I use photoshop to create sketches for my world, the brown block is a normal block and the red block is the water which the player died once he collides with it. After creating the sketch, I apply it to my code my creating a list of numbers which wil be use to create the world.

Background pattern

Description automatically generated

Here is my list for each world which can be accessed by using indexing. Each number inside the list represent 1 block in the world. The world width and height are 20 blocks x 20 blocks so there are a total of 400 blocks per world, 400 numbers.

Text

Description automatically generated

Here are the example of the if statement which is used to convert the number in the world\_data list into the image of the block. The tile are numbered from 1-9 with different kind of images and also I add a class for the enemy and water. By appending the tile to the self.tile\_list, it will collect all the images.Graphical user interface, text

Description automatically generated

By using this function screen.blit, it will show the images onto the screen.

1. **Player**

Text

Description automatically generated A picture containing qr code

Description automatically generated

Text

Description automatically generated

To create the player, I create a player class which contain an innit function. The function is to state the variables for the player and import the images for the player animation. Self.image is the image of the player which will be update every time the player moves. The for loop inside the function is to import the images from the img file to create the animation and append it to the list so when the player move to the left it use the self.image\_left list and when to the right, it used the self.image\_right list. transform.flip is to flip the image from left to right. I also added a dead image which will be drawn to the screen when the player collides with an enemy or water.

**Player’s movement**

Text

Description automatically generatedText

Description automatically generated

To move a player, pygame.key.get\_pressed() is needed so the when the user press the keyboard buttons, the code will run / do something. There are if condition which is used when the player press space, left button, right button.

Vel\_y = -15 means to move the player above by 15 pixels. Going up uses “-“ and down uses “+”. The same goes to left and right. Dx is the position of the player, by minusing it with 5 the player goes to the left and adding 5 the player goes to the right. For the final if statement, if the player stop after going to the left, by using the self.image\_right[index], images can be accessed to create the player heading to the left and the same thing goes after the player moves to the right. For the player to jump and land back to the ground, gravity is needed. Self.vel\_y is the gravity which will increase and when it is above ten it will not increase again.so after the player press space, self.vel\_y will be -15 and directly added 1 until it equals to 10. In the if statement the self.vel\_y equals to -15 which means the player is going up.

**Player Animation**

A screenshot of a computer

Description automatically generated with medium confidence

By using the if statement when the player is moving, it will directly check to the if statement inside it, which is self.direction == 1 an self.direction == -1, if self.direction equals to 1, it means the player is moving to the left so the self.image would be the image moving to the left, the same goes to the right.

**When the player dies**

Text

Description automatically generated

**When the game runs (part of Main code)**

Text

Description automatically generated

The if world\_num = 1 is to show which level the player is in, 1 means that the player is in level 1. In different levels, the collision for the player can be changed but in this program the content of the if statement for each level are the same. The way to check if the player collides or not is by using the pygame.sprite.spritecollide(). So, the if statement is to check which type of object do the player collides. If the player collides with water / slug, the game\_over will be -1 which makes the player to stop moving and died. If the player collides with the door, game\_over == 1 which means the user has finished the level and will be directed to the main cover of the screen.

**Player Died**

Text

Description automatically generated

So once the player died which means game\_over == -1. The image of the player will change into the dead image which is the graveyard. Screen.blit is to draw the player unto the screen.

**Player Collision**

A picture containing chart

Description automatically generated

**The rectangle for each object is used so the player will not fall of the screen**

By using the rectangle tile for each tile and the player it is used so each object will not overlap when colliding.

Text

Description automatically generated

By using collidirect, python can directly check if the player and the object tile are colliding so they will not overlap. Self.rect.x and self.rect.y are the coordinates for the object and self.width and height is the width and height of the object itself.

The if condition stated are used when the player is moving left / right and jump /fall.

1. **Classes**

A picture containing text

Description automatically generated

There are 11 classes with 8 different classes. In different levels the world, enemy, and water class are different. It is to differentiate the rectangle object created behind the screen for each level.

Text

Description automatically generated

**The button class** is used to create the button which can be pressed then give an output. The button class is to create a restart, start, and edit button. The player class is as explained as in Player part (number 5).

**The world class** is to create the world by using images as tiles. By using the for loop, the list of worlds which was create can be opened and read. The if statements for each tile is to identify the numbers which will be changed into the tile when the world is created. The coding for each if statements are almost the same, first the image got transformed as the same size of the tile which was stated at the beginning. Then create the image rect to create the player collision which the object so they won’t crash. Then append the tile into the tile list which later will be used to create the world.

Tile 1 is dirt, Tile 2 is dirt with a grass above it, tile 3 is water, tile 4 is enemy, tile 5 is an exit door, tile 6 is a trunk(decorations), Tile 7 is hedge, Tile 8 and 9 acts as a decoration as well. Water and enemy don’t need much code as they already have their own class with their own characteristics. So, by putting the class and appending them to the list, the enemy and water can already be created.

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

The images above are the Code for World class

World class 1 and 2 are both the same content, the user can also change the characteristic of each world to create unique levels but for this time, both are the same.

At the very bottom of the class, there is a function called draw which is to draw the tile into the screen by opening the list.

**Enemy and Water Class**

Text

Description automatically generated

Text

Description automatically generatedA picture containing text, clock

Description automatically generated

The enemy and water class are similar except that the water class does not moves so the water class does not need the update function. The update function is to update the enemy position which will move 1 block to the left and 1 block to the right. By using the if statement the enemy will move 1 block to the right, then 2 block the left, to block to the right and doesn’t stop. The transform.flip on the enemy class in the update function is used to flip the image when it heads different direction(head left when moving to the left and head right when moving to the right).

Text

Description automatically generatedA picture containing qr code

Description automatically generated

The exit class is simple only contains the image load and transform it into the size of a tile. The self rect is always needed when the player collides with it

1. **Button**

There’s a button class which was used to create the button

Text

Description automatically generated

By using the class, various button can be created. The button can be spawned in the screen by using the draw function. For example when drawing restart\_button, the draw function can be used(restart\_button.draw). This code is use in the main menu, when the code is running and when the player died (reset button).

Text

Description automatically generated

A picture containing text, toy

Description automatically generatedLogo

Description automatically generated

The button used in the main menu

Text

Description automatically generated

When the player died, a respawn button will appear for the user to pressed. When the button is pressed the position of the player is reset to the first place they spawned.

1. **Sound Effects and background sound.**

Text

Description automatically generated

by importing the mixer module, music and sound effect can be imported into this code. The background music which was played when the game is running is created by the music.play function. The 5000 inside the mixer.sound is to create the sound a little fade so it doesn’t directly appear in the game. The music.play will be played as long as the game is running.

The jump\_fx is the variable to store the sound when the player jumps. And by using the mixer.sound function the file could be stored inside a variable. The same thing as the game\_over which is used to store the sound effect when the player died

The volume for each sound effect can also be set by using the set\_volume function.

**Using it into the game**

A screenshot of a computer

Description automatically generated with medium confidence

Inside the player class, when the player jump the sound effect of jumping is played

Text

Description automatically generated

For the update function inside the player class, when the user collides with an enemy or water the game-over sound effect is played